Obligations with deadlines and maintained interdictions in privacy regulation frameworks

Guillaume Piolle
Université Joseph Fourier - Laboratoire d'Informatique de Grenoble

Yves Demazeau
CNRS - Laboratoire d'Informatique de Grenoble
46 avenue Félix Viallet
F-38031 Grenoble Cedex, France
guillaume.piolle@imag.fr – yves.demazeau@imag.fr

Abstract

We aim at providing artificial agents with logical tools to reason specifically on privacy-related regulations, in order to comply with them. In order to express these regulations, we propose a deontic and temporal logic based on predicates dealing with personal data management. Using an example, we show the need for specific operators to express obligations with deadlines and maintained interdictions. We define a set of eight specific requirements for such operators, we evaluate the existing proposals with respect to these requirements and we adapt our own ones, to better suit to our formalism.

1. Introduction

Commercial service providers tend to put a stress everyday more significant on personalization, thus collecting more and more personal data from users. In consequence, they have to provide them with guarantees on the protection of their private information. The users' concern for privacy has indeed grown jointly with the thirst for personalization. Intelligent agents aiming at interfacing their human user with distant or distributed services should then be provided with tools for protecting private information, in application of the privacy regulations relevant to their execution context.

In the general domain of privacy, research has first focused on the so-called *Privacy-Enhancing Technologies* (PETs), the technical means used to protect individuals' information. The ISO, for instance, gave in 1999 a set of "common criteria" to be met by such systems, in terms of technical features [8]. In 2006, Yves Deswarte classified existing Internet-oriented PETs with respect to their aims (identity management, IP anonymization...) [6] while we proposed the same year a classification of privacy-related regulations along six dimensions (user information, user consent, data update and retractation, process justification, data retention and data forwarding) [11]. Regarding the needed reasoning means on these regulations, much work has been done on the representation of security policies using deontic logic. One could cite for instance the early works of Ortalo [9] or the proposals of Cholvy and Cuppens on the consistency of such policies [3]. These works seem to provide us with means to represent the normative context of an agent. So far, none of the logic-based proposals include privacy-specific reasoning features, even though some of them deal with closely related fields like information exchange policies [4].

The need for temporal dynamism is particularly prominent in privacy regulations, which often deal with notions such as durations and deadlines. Such norms may state, for instance, that a piece of information should be deleted before a given date or after a given duration, or that user information must take place at least N days before her data are used. Techniques for dealing jointly with deontic and temporal notions has often been studied, most notably by Lennart Åqvist [1], and some authors have been working "in the abstract" on the general notion of deadlines in deontic obligations [7, 5, 2]. This notion is actually crucial to the representation of privacy norms, and we need to thoroughly analyse how these contributions can help us in our work. What we propose here is to build a deontic and temporal logic dedicated to the representation of privacy regulations and to make sure that it is expressive enough for notions such as obligations with deadlines and maintained interdictions.

In the following section, we present the *Deontic Logic*

for Privacy (DLP), designed to deal with privacy-specific notions. In section 3, we specify the requirements for an "obligation with deadlines" operator, see how three operators already proposed match these requirements once translated into DLP, and how a more suited one can be built on this basis. We also show how the notion of interdictions maintained over time can be expressed in DLP. In section 4, we use these operators to translate a fictional privacy regulation into a DLP norm. In the following, we will consider a **service agent**, in charge of a process (or service) involving personal information, and a **user agent**, willing to obtain that service and responsible for the personal information of its human owner.

2 The DLP logic

The DLP logic is a normal deontic and (linear) temporal language, the corresponding modalities being applied on a base language $\mathcal{L}_{DLP}$ designed to represent information and actions about personal data usage and protection.

2.1 Privacy-related information

In $\mathcal{L}_{DLP}$, processes making use of personal information are identified through a unique ID. Processes have one or more ACTIONTYPES to define their nature. Each process is related to a set of personal information chunks, each of them being identified by a DATAID (local to the process) and characterized by one or more DATATYPES. In the current version of the language, ACTIONTYPES and DATATYPES take their values in the various sets and type trees defined by the P3P standard schemata [14]. The language is based on three sets of predicates, on which the connectives of propositional logic ($\neg, \vee$) are applied[1] :

- *Informative predicates*, designed to represent the service agent informing a user agent with respect to a given process. Information can be about the type of the process (*informActionType* predicate), the potential recipients of a given piece of information (*informForward*), the information retention time (*informDuration*) or the agent to contact for further data update or deletion requests (*informContact*).
- *Performative predicates*, designed to represent some actions taken by the agents. Service agents can request for a piece of data in order to provide the service (*request*), ask for the consent of the user agent with respect to a process (*requestConsent*), execute the process (*perform*), update an information (*update*), delete it (*forget*) or forward it to another service agent (*forward*). User agents can consent to a process (*consent*), send a piece of information (*tell*), request a data update (*updateRequest*) or a data deletion (*forgetRequest*).
- *State predicates*, used to memorize information in the form of relational tuples. With these predicates, processes are associated to their ACTIONTYPES (*actionType*), the service agent responsible for them (*responsible*) and the associated contact agent (*contact*). Pieces of information are linked to their DATATYPES (*dataType*), their owner (*owner*), their potential recipients (*forwardList*) and their registered retention duration (*duration*).

Any instance of these predicates can be true or false at a given instant in the timeflow, thus allowing a precise description of past and planned events. One must notice though that $\mathcal{L}_{DLP}$ is not an agent communication language. Its only aim is to internally represent the information about the speech acts taken by the agents, not to embody them.

For each predicate in the language, each argument takes its value in a domain which is a finite or countable set. Therefore, the set of all possible $\mathcal{L}_{DLP}$ predicate instances is countable and can be mapped onto a set of (virtual) propositional atoms. In the case one of the arguments is not bound in an expression, then the predicate represents a propositional schema (like axiomatic schemata), covering a whole set of propositional atoms. One should note that there are no explicitly quantified formulae in the language. This is why, even though we use the form of predicates to express our information, we can consider a modal logic based on $\mathcal{L}_{DLP}$ as a propositional modal logic.

2.2 DLP Syntax

DLP is a language where the obligation modality Ob of Standard Deontic Logic (SDL [13]) is freely mixed with temporal operators from Propositional Linear-time Temporal Logic (PLTL [12]). The well-formed formulae φ of the DLP language are defined as follows, where p is a $\mathcal{L}_{DLP}$ proposition[2]:

$$\begin{aligned}\varphi = \quad & p \mid \varphi \text{ “}\vee\text{” } \varphi \mid \text{“}\neg\text{” } \varphi \mid \text{“}Ob\text{” } \varphi \\ & \varphi \text{“}\mathcal{U}\text{” } \varphi \mid \text{“}H\text{” } \varphi \mid \text{“}X^{-1}\text{” } \varphi \,;\end{aligned} \qquad \text{(DLP-1)}$$

We have chosen a temporal language allowing much expressivity on the future, through a strict "until" operator $\mathcal{U}$. A loose until $\mathcal{U}^-$ is defined as well, including the present (DLP-2). Since this logic is to be used by an agent to reason

[1] For page limitation reasons, we do not give the full argument list and usage of each predicate.

[2] The logic presented here is actually a simplified version of DLP. The complete version makes use of a class of obligation modalities Ob^{ν}_{a} (rather than the single one Ob used here), differentiated by agent and by normative authority. This is to allow normative conflict representation and arbitration, as it has been shown in a previous communication [10], but does not bring anything essential to the problem addressed here.

about its own behaviour and the plans it chooses to execute, more expressivity is needed in the future than in the past, where a "since" operator has not proved essential. We thus rely on an universal modality H on the past, and a X^{-1} operator pointing to the previous time step (DLP is to be interpreted over non-dense timeflows). Its symmetric operator in the future is a "next" modality X, defined as an abbreviation ($X\varphi \stackrel{def}{=} \bot \mathcal{U} \varphi$). We will use the usual temporal abbreviations G (universality in the future), F (existential dual of G) and P (existential dual of H), none of them including the present. We define their loose counterparts G^-, H^-, F^-, P^- including the present.

$$\varphi \mathcal{U}^- \psi \stackrel{def}{=} \psi \vee (\varphi \wedge \varphi \mathcal{U} \psi) \quad \text{(DLP-2)}$$

$$F\varphi \stackrel{def}{=} \top \mathcal{U} \varphi \quad \text{(DLP-3)}$$

$$G\varphi \stackrel{def}{=} \neg F \neg \varphi \quad \text{(DLP-4)}$$

$$P\varphi \stackrel{def}{=} \neg H \neg \varphi \quad \text{(DLP-5)}$$

We also define a class of indexed X^i operators, based on X and X^{-1}, which can be used to travel step-wise along a timeflow:

$$\begin{cases} X^0\varphi \stackrel{def}{=} \varphi \\ \forall n \in \mathbb{Z}_+^*, X^n\varphi \stackrel{def}{=} XX^{n-1}\varphi \\ \forall n \in \mathbb{Z}_-^*, X^n\varphi \stackrel{def}{=} X^{-1}X^{n+1}\varphi \end{cases} \quad \text{(DLP-6)}$$

2.3 DLP Semantics

DLP formulae are interpreted over Kripke-like bi-dimensional structures which we call DLP models.

Definition 1 (DLP model) *A DLP model $\mathcal{M}$ is a set $\{\mathcal{H}, \mathcal{T}, \rightsquigarrow, \mathbb{O}, v\}$ where:*

- *$\mathcal{H}$ is a countable set of objects ($h, h', h''... \in \mathcal{H}$) called histories;*
- *$\mathcal{T}$ is a countableset of objects ($t, t', t''... \in \mathcal{T}$) called dates. A couple from $\mathcal{I} = \mathcal{H} \times \mathcal{T}$ ($i, i', i''... \in \mathcal{I}$) is called an instant (or a world, for DLP interpretation is based on them);*
- *$\mathbb{O}$ is a (serial) binary relation among instants ($\mathbb{O} \subset \mathcal{I} \times \mathcal{I}$) called the deontic accessibility relation;*
- *$\rightsquigarrow$ is a binary relation (serial, left-unbounded, linear in both directions) among dates ($\rightsquigarrow \subset \mathcal{T} \times \mathcal{T}$) called the temporal accessibility relation;*
- *v is an application from $\mathcal{L}_{DLP}$ to $\wp(\mathcal{I})$ called the valuation function.*

A DLP model is then a bi-dimensional grid where an instant i is the point defined by a couple of coordinates (h, t). A particular date t_0 can be identified as an arbitrary reference point for the timeflow. The truth of $\mathcal{L}_{DLP}$ propositions and DLP formulae is defined for instants, which are the elementary worlds of the structure. For each instant $i = (h, t)$, the equivalent notations $\mathcal{M}, h, t \models \varphi$ and $\mathcal{M}, i \models \varphi$ read, respectively, "φ is true at date t in history h of model $\mathcal{M}$" and "φ is true at instant i of model $\mathcal{M}$". We define the relation $<$ (resp. $\leq$) among dates as the transitive (resp. transitive and reflexive) closure of $\rightsquigarrow$. DLP formulae are interpreted over DLP structures as follows (with $p \in \mathcal{L}_{DLP}, \varphi, \psi \in DLP$):

$$\mathcal{M}, h, t \models p \text{ iff } (h, t) \in v(p) \quad \text{(DLP-7)}$$

$$\mathcal{M}, h, t \models \neg\varphi \text{ iff } \mathcal{M}, h, t \not\models \varphi \quad \text{(DLP-8)}$$

$$\mathcal{M}, h, t \models \varphi \vee \psi \text{ iff } (\mathcal{M}, h, t \models \varphi \text{ or } \mathcal{M}, h, t \models \psi) \quad \text{(DLP-9)}$$

$$\mathcal{M}, i \models Ob\,\varphi \text{ iff } (\forall i' \in \mathcal{I}, \text{ if } i\mathbb{O}i' \text{ then } \mathcal{M}, i' \models \varphi) \quad \text{(DLP-10)}$$

Thus, a deontic obligation on a formula at a given date of a given history means that this formula is true at all instants accessible via $\mathbb{O}$. One could note that $\mathbb{O}$ is not a relation among histories: a deontically acceptable alternative is a date in a history, and not the history as a whole. This gives a better granularity to the notion of norm violation. The following formulae give the semantics of the temporal operators: $\varphi \mathcal{U} \psi$ is true at a given date of a given history if and only if, in the same history, ψ occurs at some point in the (strict) future, and φ is true from the next instant (tomorrow) until then. $H\varphi$ is true at an instant if and only if φ has been true at every past instant in the same history, and $X^{-1}\varphi$ is true at an instant if and only if φ was true at the previous instant in the same history. It is then straightforward to check that the semantics of temporal abbreviations is defined as usual.

$$\mathcal{M}, h, t \models \varphi \mathcal{U} \psi \text{ iff } \exists t' \in \mathcal{T}, \begin{cases} t < t', \\ \mathcal{M}, h, t' \models \psi \\ \forall t'' \in \mathcal{T}, \text{ if } t < t'' < t' \text{ then } \mathcal{M}, h, t' \models \varphi \end{cases} \quad \text{(DLP-11)}$$

$$\mathcal{M}, h, t \models H\varphi \text{ iff } \forall t' \in \mathcal{T}, \text{ if } t' < t \text{ then } \mathcal{M}, h, t' \models \varphi \quad \text{(DLP-12)}$$

$$\mathcal{M}, h, t \models X^{-1}\varphi \text{ iff } \forall t' \in \mathcal{T}, \text{ if } t' \rightsquigarrow t \text{ then } \mathcal{M}, h, t' \models \varphi \quad \text{(DLP-13)}$$

3 Dated deontic operators in DLP

We will now examine how obligations with deadlines and maintained interdictions can be defined in DLP. As an example, we will consider the following (fictional) norm: *users must be informed at least one week in advance of the nature of any process involving their personal information*. This sentence looks rather simple, and not uncommon. It could be part of a corporate regulation, for instance. However, its interpretation is not so straightforward, and could benefit from the notions of obligations with deadlines and maintained interdictions. Indeed, an agent could consider this norm from two different points of view. On the one hand, if a given user has not been informed so far about a given process, then there is a standing interdiction (maintained for one week) to perform that process. On the other hand, if the agent has planned to perform a process more than one week in the future, then it has the obligation to inform the user before a date situated one week before the process. The regulation will then be translated into (a conjunction of) two partially redundant DLP formulae, representing the two sides of the problem[3]. This example norm is pretty rich, showing the need for the dated operators of obligations with deadlines and maintained obligations. Other more intuitive or more common example norms may have resulted in simpler interpretation cases.

In the following, we will consider the building of an operator $Ob(\varphi, \delta)$ intended to mean: "it is obligated that φ becomes true strictly before δ". Later on we will see how to construct its deontic counterpart, the maintained obligation operator $For(\varphi, \delta)$. We define $violOb(\varphi, \delta)$ and $violFor(\varphi, \delta)$ as the respective violations of these deontic operators.

3.1 Requirements

In order to represent deadlines, we add a special kind of propositions to the language, the *dated propositions*. The additional predicate *date*(δ) means that δ is a dated proposition. They are defined as propositions occuring once and only once in the timeflow. Furthermore, for practical reasons, we consider that the dated proposition δ_i correspond to the date t_i ($\forall i \in \mathbb{Z}, \forall h, \mathcal{M}, h, t_i \models \delta_i$).

$$date(\delta) \stackrel{def}{=} \begin{cases} \delta \wedge G\neg\delta \wedge H\neg\delta \\ \vee F(\delta \wedge G\neg\delta \wedge H\neg\delta) \\ \vee P(\delta \wedge G\neg\delta \wedge H\neg\delta) \end{cases} \quad \text{(DLP-14)}$$

We introduce and discuss eight *requirements* that an operator for obligations with deadlines must match, in order to properly represent the notion in DLP logic.

1. *Deadlines must be dated propositions*: for a deadline to bear a meaning, it must be defined in a clear and distinct fashion. It must exist and be unique in the timeflow. Our dated propositions have been defined for that purpose.

2. *The deadline must be in the future*: it is obvious for a human that a present or past deadline does not leave any choice or liberty to plan actions.

3. *Obligations on $\top$ must be tautologies*: we introduce this principle in coherence with the immediate deontic obligation, for which $Ob\ \top$ is a theorem. In the same way, $Ob(\top, \delta)$ should be a theorem.

4. *Obligations on $\bot$ should be antilogies*: for the same reasons, $\neg\ Ob(\bot, \delta)$ should be a theorem.

5. *Violated obligations should be dropped after the deadline*: this choice is rather philosophical in nature, and subject to debate (although it is a consequence of point 1). We think that if the agent could not cope with the obligation on time, then it is too late and the violation must be simply accepted. This principle is not an opposition to the existence of eventual contrary-to-duty norms attached to dated obligations.

6. *Violations should be punctual*: we prefer to think of violations as events, not states. It gives a better expressivity and is allowed by the fact that we can reason on the past. Again, this point is specific to the chosen formalism and subject to debate.

7. *The propagation principle must be respected*: it says that the obligation with deadline must be maintained from instant to instant, until the obligation is respected or the deadline is reached.

$$Ob(\varphi, \delta) \wedge \neg(\varphi \vee \delta) \wedge \neg X\delta \rightarrow XOb(\varphi, \delta) \quad \text{(P)}$$

8. *The monotony principle must be respected*: it says that if there is an obligation with a deadline, then an obligation with a further deadline can be derived (M).

$$Ob(\varphi, \delta) \wedge F(\delta \wedge F^{-}\delta') \rightarrow Ob(\varphi, \delta') \quad \text{(M)}$$

The first six *requirements* have already been pointed out as choice points and discussed by Dignum [7]. However, due to their different goals and formalisms, their opinion differ on some aspects. For instance, since they do not reason on past, they define violations as states, not events. Furthermore, they reject obligations on $\top$ since they are not coherent with the STIT operator they use. On some other points, they do not make a clear choice and leave the final

[3] There are actually other cases to be examined by the agent, like when a process is planned less than one week in the future and the user has not been informed. In this cases, the unavoidable violation is triggered by the first form of the norm. The reader can check that the remaining possibilities are covered by the two norm descriptions we have given.

decision to the reader, depending on her needs. Propagation and monotony *requirements* have been introduced by Brunel [2], and we have adapted them to our more strict interpretation of the notion of deadline.

3.2 Evaluating some existing proposals

Several works have been done on obligations with deadlines, and we evaluate three of them. We choose to examine three proposals made by Dignum [7], Demolombe [5] and Brunel [2]. We will adapt them to our logical formalism (we will note their proposed operators, respectively, Ob^a, Ob^b and Ob^c)[4] and see how these adaptations satisfy our requirements.

The operator Ob^a by Dignum is defined jointly with the notion of violation, and does not rely on a real deontic logic [7]. Here, an obligation with a deadline is characterized by the absence of violation before the deadline, and the maintaining of a violation, or its negation, after the deadline, depending on whether the obligation has been fulfilled on time or not. Because of its strictly temporal definition, such obligations can be derived whenever they seem to be respected. This is quite an undesirable feature for building norm-directed agents. Further more, it is not monotonic at all: there are several cases where an obligation with a further deadline cannot be derived[5]. Deadlines with a value of $\top$ can be defined, resulting in an immediate obligation.

The Ob^b operator, adapted from Demolombe [5], translates better in a deontic and temporal logic. An obligation with a deadline is now the maintaining (until the deadline or the fulfilment of the obligation) of an obligation of anteriority between the target formula and the deadline. Propagation is here built-in in the operator, in the form of immediate obligations allowing us to derive dated obligations. Regarding monotony, if an obligation $Ob^b(\varphi, \delta)$ is fulfilled on time, then it is possible to check that $Ob^b(\varphi, \delta')$ is true for any future δ'. However, if it is violated, it is not possible to derive an obligation with a further deadline.We will call this semi-monotony. It is interesting to see that this property has a nice side-effect: it prevents us from deriving multiple violations for the same initial obligation, whereas it still allows monotony if the obligation is fulfilled.

The Ob^c operator is based on two different dated obligations [2]. The first one, $Ob^{c'}$, is used when the obligation is initially activated, and does not allow propagation (although it is monotonic). Brunel, unlike precedent authors, reason on delays of k time steps ($k \in \mathbb{N}$) rather than on deadlines. The final operator has the following meaning: an obligation with a deadline is active at the present time if there was in the past an obligation (of type $Ob^{c'}$, and not fulfilled yet) on φ with a deadline not reached already, and also that there was no obligation on the same formula with a shorter deadline. This more complex operator deals efficiently with propagation and is able to express subtle nuances of dated obligations. Yet, it has several noticeable drawbacks in our application context. First, it introduces new quantifiers on delays (when Ob^b relies on existing temporal quantifiers, for instance). For this reason it is not directly expressible in our modal language. Plus, the operator does not respect the monotony principle, although it deals properly with other requirements, by reasoning on delays rather than on deadlines.

Table 1 summarizes how the three operators match our eight requirements. An empty square means that the requirement is only partially satisfied, or under some conditions.

Requirement	1	2	3	4	5	6	7	8
Ob^a		□	□		■		■	
Ob^b			■	□	■	□	■	□
Ob^c	■	■		■	■	□	■	

Table 1. Evaluation of existing operators

3.3 An operator adapted to DLP

Even though Ob^c satisfies more of our requirements, both Ob^a and Ob^c imply too significant extensions to our logic to be directly usable (namely, the non-deontic nature of Ob^a and the explicit quantification on durations in Ob^c). The operator Ob^b seems to be a better trade-off, being much more compatible with DLP formalism. It can be improved by the use of dated propositions, so that it satisfies our requirements in a better way. For the reason we have given, we will accept semi-monotony rather than strict monotony for our operator, which is now defined (along with its punctual violation, adapted to the persistent one proposed by Demolombe) as follows:

$$Ob(\varphi, \delta) \stackrel{def}{=} \begin{cases} date(\delta) \\ F\delta \\ Ob(F^-(\varphi \wedge F\delta))\, \mathcal{U}^-\, (\varphi \vee \delta) \end{cases} \quad \text{(def-1)}$$

$$violOb(\varphi, \delta) \stackrel{def}{=} \begin{cases} date(\delta) \\ \delta \\ P(Ob(\varphi, \delta) \wedge \neg\varphi\, \mathcal{U}^-\delta) \end{cases} \quad \text{(def-2)}$$

It is now easy to show that this adapted operator satisfies the eight requirements (with monotony replaced by semi-monotony) defined in section 3.1.

[4] Operators appear in a different formalism in the original papers, and have slightly different properties in a different context. Ob^c, for instance, is originally expressed using (unnecessary) extensions of the temporal language, which we have translated into standard LTL operators.

[5] Namely, if the obligation is fulfilled and then another obligation on the same formula is violated after the first deadline, or if the obligation is violated and then φ becomes true after the deadline.

Theorem 3.1 *The operator $Ob(\varphi, \delta)$ and its associated violation $violOb(\varphi, \delta)$ have the following properties:*

1. *Deadlines are dated propositions (they occur once and only once in the timeflow);*
2. *Deadlines are always in the strict future;*
3. *$Ob(\top, \delta)$ can be derived for each properly defined deadline δ;*
4. *$\neg Ob(\bot, \delta)$ is a theorem;*
5. *Obligations are dropped after the deadline;*
6. *Violations are punctual:*
$$violOb(\varphi, \delta) \rightarrow \left\{ \begin{array}{l} G\neg violOb(\varphi, \delta) \\ H\neg violOb(\varphi, \delta) \end{array} \right.$$
7. *Propagation principle is respected;*
8. *Semi-monotony is respected:*
$$\left. \begin{array}{l} Ob(\varphi, \delta) \wedge date(\delta') \\ \wedge F(\delta \wedge F^{-}\delta') \wedge \neg(\neg\varphi\, \mathcal{U}^{-}\, \delta) \end{array} \right\} \rightarrow Ob(\varphi, \delta')$$

Properties 1 and 2: trivial, by definition of the operator. *Property 3:* once we have $date(\delta)$ and $F\delta$, $Ob(\top, \delta)$ reduces to $Ob(F^{-}(\top \wedge F\delta))\, \mathcal{U}^{-}\, (\top)$, then to $\top\, \mathcal{U}^{-}\, \top$, which is $\top$. *Property 4:* $Ob(\bot, \delta)$ implies $Ob(F^{-}(\bot))\, \mathcal{U}^{-}\, (\delta)$. $F^{-}(\bot)$ implies $F(\bot)$, then $\bot$. By the D axiom of SDL obligations, the original expression then boils down to $\bot\, \mathcal{U}^{-}\, \delta$ and then to $\bot$. *Property 5:* $Ob(\varphi, \delta) \rightarrow F\delta$, so $F^{-}Ob(\varphi, \delta) \rightarrow F\delta$, then $\neg G^{-}\neg Ob(\varphi, \delta) \rightarrow F\delta$. $F\delta$ implies $\neg\delta$, so $(\neg G^{-}\neg Ob(\varphi, \delta)) \rightarrow \neg\delta$. *Modus tollens* then leads us to the result. *Property 6:* $violOb(\varphi, \delta)$ implies δ, so it can be true only once in the timeflow. *Property 7:* the antecedent of the proposition allows us to derive $Xdate(\delta)$ and $XF\delta$ (through $\neg X\delta \wedge F\delta$). $\neg(\varphi \vee \delta)$ tells us that the deadline is not reached yet, so the whole $\mathcal{U}^{-}$ expression will be true at next time step. By conjunction of these points, we get $XOb(\varphi, \delta)$. *Property 8:* the antecedent of the proposition gives us $date(\delta')$ and $F\delta'$. $\neg(\neg\varphi\, \mathcal{U}^{-}\, \delta)$ implies that φ will be true before δ, so before δ'. Therefore, $Ob(F^{-}(\varphi \wedge F\delta))\, \mathcal{U}^{-}\, (\varphi \vee \delta)$ implies $Ob(F^{-}(\varphi \wedge F\delta))\, \mathcal{U}^{-}\, (\varphi \vee \delta')$. On the other hand, $(\varphi \wedge F\delta)$ implies $(\varphi \wedge F\delta')$, therefore by the normal character of modalities we get $Ob(F^{-}(\varphi \wedge F\delta'))\, \mathcal{U}^{-}\, (\varphi \vee \delta')$. The antecedent therefore allows us to derive $Ob(\varphi, \delta')$. □

3.4 Maintained obligations

It is much easier to defined an operator with good properties for maintained interdiction than for obligations with deadlines. Our operator and its associated violations are defined by the formulae (def-3) and (def-4). It is easy to show that this operator has eight properties corresponding to the ones we have seen (where principle 5 would be the maintaining of interdictions, even violated, until the deadline, and monotony would be the ability to derive an interdiction with a sooner deadline).

$$For(\varphi, \delta) \stackrel{def}{=} \left\{ \begin{array}{l} date(\delta) \wedge F\delta \\ (For\ \varphi)\, \mathcal{U}^{-}\, \delta \end{array} \right. \quad \text{(def-3)}$$

$$violFor(\varphi, \delta) \stackrel{def}{=} \left\{ \begin{array}{l} date(\delta) \wedge F\delta \wedge \varphi \\ P^{-}For(\varphi, \delta) \end{array} \right. \quad \text{(def-4)}$$

4 Application to privacy norms

Now that we have two suitable operators for obligations with deadlines and maintained obligations, we can translate in DLP logic the interpretations we have made of the fictional norm presented in part 3. Formula (ex-1) formally says that if there is an identified type for a given process, and this process uses given data, and the owner of these data has not been informed so far, and the date δ in situated one week in the future, then there is a maintained interdiction to perform the process until δ. Formula (ex-2) says that if δ is in the future and one week before a planned process, and the owner has not been informed yet, then there is an obligation to inform her before δ.

$$\left. \begin{array}{l} \textit{actionType}(\text{ProcessID}, \text{ActionType}) \\ H\neg \textit{informActionType}(\texttt{self}, \text{User}, \\ \quad \text{ProcessID}, \text{ActionType}) \\ \textit{owner}(\text{ProcessID}, \text{DataID}, \text{User}) \\ \textit{date}(\delta) \wedge X^{7*24}\delta \end{array} \right\} \rightarrow For(\textit{perform}(\texttt{self}, \text{ProcessID}), \delta) \quad \text{(ex-1)}$$

$$\left. \begin{array}{l} \textit{actionType}(\text{ProcessID}, \text{ActionType}) \\ \textit{date}(\delta) \\ F(\delta \wedge X^{7*24-1}\textit{perform}(\texttt{self}, \text{ProcessID})) \\ H\neg \textit{informActionType}(\texttt{self}, \text{User}, \\ \quad \text{ProcessID}, \text{ActionType}) \\ \textit{owner}(\text{ProcessID}, \text{DataID}, \text{User}) \end{array} \right\} \rightarrow Ob(\textit{informActionType}(\texttt{self}, \text{User}, \text{ProcessID}, \text{ActionType}), \delta) \quad \text{(ex-2)}$$

If norm (ex-1) is activated (i.e. if its antecedent is verified), then the agent sees that any occurence of ProcessID before δ would result in a violation. A privacy-aware agent willing to comply with enacted regulations would then refrain from performing this action (by generating a BDI desire about it for instance, or by postponing any undertaken plan or intention involving ProcessID). The monotony principle here makes it easier to develop reasoning means to effectively derive short-term and even immediate interdictions on the basis of a maintained interdiction.

If norm (ex-2) is activated, then the agent knows that it has to inform the user before a given and well-known date δ. The propagation principle (and basically the structure of the operator, ensuring the utterance of an immediate obligation at each step of the period) provides a direct way to make the agent feel "concerned" already by this obligation (thus triggering planning mechanisms) rather than postponing the interpretation of the norm.

5 Conclusion and future work

In this paper we have proposed a language, based on temporal deontic logic, to represent and reason on privacy-related norms. We have pointed out the need for a specific operator dealing with obligation with deadlines. We have defined eight requirements that such an operator should match in order to bear the right intuitive meaning in our formalism. We have examined three proposed operators for operations with deadlines, ported them onto our logic, and observed that none of them met all our requirements. We have slightly adapted the most suited of these operators to satisfy our requirements and designed another one for dealing with maintained obligations. Finally, we have used these operator in the translation of an example regulation in our logic.

It is interesting to point out that the translation of privacy regulations from natural language to formal expressions cannot be done without a human interpretation phase, and not only because of the natural language processing problems. In the context of artificial agents, it would then be necessary to develop norm designing tools constraining the form of the regulations defined by a human user.

Current work involves finishing off the axiomatization of DLP logic (notably, there is a debate over temporal/deontic converse axioms, which could be an obstacle to the dynamism of the norm base) and characterizing its complexity. The DLP normative conflict detection and resolution engine based on our previous work [10] is to be adapted in order to take into account obligations with deadlines and maintained obligations. Further on, we plan to formally evaluate the characteristics and performance of the implementation and compare them to the formal model of DLP logic as well as to other possible implementations. DLP engines will then be integrated in the cognitive layers of "privacy-aware" software agents. These agents are then meant to be used as personal assistants or as automatic service providers in Internet-based (or other distributed) applications, thus helping human users to deal properly and more safely with their private information.

This research has been supported by the Web Intelligence project of the Rhône-Alpes region ISLE cluster.

References

[1] L. Åqvist. Combinations of tense and deontic modalities. In A. Lomuscio and D. Nute, editors, *Proceedings of the 7th International Workshop on Deontic Logic in Computer Science (DEON 2004)*, volume 3065 of *LNCS*, pages 3–28, Madeira, Portugal, May 2004. Springer.

[2] J. Brunel, J.-P. Bodeveix, and M. Filali. A state/event temporal deontic logic. In *Eighth International Workshop on Deontic Logic in Computer Science (DEON'06)*, number 4048 in LNCS, 2006.

[3] L. Cholvy and F. Cuppens. Reasoning about norms provided by conflicting regulations. In H. Prakken and P. McNamara, editors, *Norms, Logics and Information Systems: New Studies in Deontic Logic and Computer Science*, pages 247–264, Amsterdam, the Netherlands, 1998. IOS Press.

[4] L. Cholvy and S. Roussel. Raisonner avec une réglementation incomplète : le cas d'une politique d'échange d'informations. In *16e congrès francophone AFRIF-AFIA sur la Reconnaissance de Formes et l'Intelligence Artificielle (RFIA'08)*, Amiens, France, January 2008. AFRIF-AFIA.

[5] R. Demolombe. Formalisation de l'obligation de faire avec délais. In *Troisièmes journées francophones des modèles formels de l'interaction (MFI'05)*, Caen, France, may 2005.

[6] Y. Deswarte and C. Aguilar-Melchor. Current and future privacy enhancing technologies for the internet. *Annales des Télécommunications*, 61(3-4):399–417, 2006.

[7] F. Dignum, J. Broersen, V. Dignum, and J.-J. Meyer. Meeting the deadline: Why, when and how. In M. G. Hinchey, J. L. Rash, W. Truszkowski, and C. Rouff, editors, *Third International Workshop on Formal Approaches to Agent-Based Systems (FAABS'04)*, number 3228 in LNCS, pages 30–40, Greenbelt, MD, USA, april 2004. Springer Verlag.

[8] ISO/IEC. Information technology - security techniques - evaluation criteria for it security, part 2: Security functional requirements. Technical Report 15408-2, International Organization for Standardization, 1999.

[9] R. Ortalo. Using deontic logic for security policy specification. LAAS report 96380, LAAS-CNRS, Toulouse, France, october 1996.

[10] G. Piolle and Y. Demazeau. Une logique pour raisonner sur la protection des données personnelles. In *16e congrès francophone AFRIF-AFIA sur la Reconnaissance de Formes et l'Intelligence Artificielle (RFIA'08)*, Amiens, France, January 2008. AFRIF-AFIA.

[11] G. Piolle, Y. Demazeau, and J. Caelen. Privacy management in user-centred multi-agent systems. In *Proceedings of the 7th Annual International Workshop "Engineering Societies in the Agents World" (ESAW 2006)*, volume 4457/2007 of *LNCS*, pages 354–367, Dublin, Ireland, September 2006. Springer Verlag.

[12] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th IEEE Symposium on the Foundations of Computer Science (FOCS-77)*, pages 46–57, Providence, Rhode Island, USA, 1977. IEEE Computer Society Press.

[13] G. H. von Wright. Deontic logic. *Mind*, 60:1–15, 1951.

[14] W3C. Platform for Privacy Preferences specification 1.1., 2006. `http://www.w3.org/P3P/`.